Compiler Checks

Robert C. Seacord, Software Engineering Institute [vita[1]]

Copyright © 2005 Pearson Education, Inc.

2005-09-27

In a perfect world, C and C++ compilers would identify the potential for exceptional conditions to occur at runtime and provide a mechanism (such as an exception, trap, or signal handler) for applications to handle these events. Unfortunately, the world we live in is far from perfect. This article provides a brief description of some of the compiler capabilities that exist today.

Development Context

Integer operations

Technology Context

C, C++, IA-32, Win32, UNIX

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Integers in C and C++ are susceptible to overflow, sign, and truncation errors that can lead to exploitable vulnerabilities.

Description

In a perfect world, C and C++ compilers would identify the potential for exceptional conditions to occur at runtime and provide a mechanism (such as an exception, trap, or signal handler) for applications to handle these events. Unfortunately, the world we live in is far from perfect. A brief description of some of the capabilities that exist today follows.

Visual C++

The Visual C++ .NET 2003 compiler generates a compiler warning (C4244) when an integer value is assigned to a smaller integer type. At warning level 1, a warning will be issued if a value of type __int64[14] is assigned to a variable of type unsigned int. At warning level 3 and 4, a “possible loss of data” warning is issued if an integer type is converted to a smaller integer type. For example, the assignment in the following example is flagged at warning level 4:

1. daisy:274 (Seacord, Robert C.)

14. http://msdn.microsoft.com/library/en-us/vccelang/htm/basic_45.asp

```
// C4244.cpp
// compile with: /W4
int main() {
  int b = 0, c = 0;
  short a = b + c; // C4244
}
```

Visual C++ .NET 2003 also provides runtime error checks that are enabled by the /RTC flag. The /RTCc compiler flag provides a similar function to compiler warning C4244 by reporting when a value assigned to a smaller data type results in a loss of data. Visual C++ also includes a runtime_checks pragma that disables or restores the /RTC settings, but does not include flags for catching other runtime errors such as overflows. Visual C++ 2005 adds the ability to catch overflows in operator::new (and is on by default).

Runtime error checks are not valid in a release (optimized) build for performance reasons [Microsoft 03].

GCC

The gcc and g++ compilers include an -ftrapv compiler option that provides limited support for detecting signed integer exceptions at runtime. According to the gcc man page, this option “generates traps for signed overflow on addition, subtraction, and multiplication operations.” In practice, this means that the gcc compiler generates calls to existing library functions rather than generating assembler instructions to perform these arithmetic operations on signed integers. If you use this feature, make sure you are using gcc version 3.4 or later because the checks implemented by the runtime system before this version do not adequately detect all overflows and should not be trusted [Seacord 04].

References

[Microsoft 03]	C++ Compiler Options, /RTC (Run-Time Error Checks), Visual Studio .NET help system.
[Seacord 04]	Seacord, Robert C. “libgcc contains multiple flaws that allow integer type range vulnerabilities to occur at runtime” (US-CERT Vulnerability Note VU#540517). April, 2004. http://www.kb.cert.org/vuls/id/540517 (2004).

Pearson Education, Inc. Copyright

This material is excerpted from *Secure Coding in C and C++*, by Robert C. Seacord, copyright © 2006 by Pearson Education, Inc., published as a CERT® book in the SEI Series in Software Engineering. All rights reserved. It is reprinted with permission and may not be further reproduced or distributed without the prior written consent of Pearson Education, Inc.

Velden

Naam	Waarde
Copyright Holder	Pearson Education

Velden

Naam	Waarde
is-content-area-overview	false
Content Areas	Knowledge/Coding Practices
SDLC Relevance	Implementation
Workflow State	Publishable